Ben Bauer
Kevin O'Farrell

Capstone Abstract

For our capstone project, we implemented a basic distributed file system, based upon Oceanstore, a UC Berkeley project from the early 2000s. Appropriately named Puddlestore, our project is not quite as robust as its source material.

As part of the implementation, we used a Distributed Object Location and Retrieval service we worked on earlier in the semester, Tapestry, and a consensus protocol we worked on earlier in the semester, Raft.

The basic implementation of Puddlestore allows users to create, remove, read, and write from files. It also allows for creation, removal, and listing of directories. As part of our capstone implementation, we also improved upon our original Tapestry implementation, adding in Publishing Path Caching, which allows objects to be found more quickly within the file system, as well as Hash Salting and Object Re-Replication, both of which make the system as a whole more fault tolerant and reliable.

In the basic implementation of Puddlestore, when a Tapestry node publishes an object, it routes along a path of other Tapestry nodes, ultimately arriving at a "root" node. This root node then keeps track of the original node's location. When another node wants to retrieve the object, they also route to the root node, from there acquire the location of the node publishing the object, and then query it from the object. The addition of Publishing Path Caching speeds up this object retrieval process. In addition to the root node knowing the publishing node's location, every other node along the path from the publishing node to the root node is also made aware. When a node tries to route to the root node of the object, their search can be stopped early if they run into one of these path nodes.

In the basic implementation of Puddlestore, as explained before, when a Tapestry node publishes an object, it informs a root node of its location. This leads to a potential issue where the root node may be unreliable or may crash, leading to difficulty retrieving the published object. To combat this issue, we implemented Hash Salting. When a node publishes an object, rather than routing to a single root node, we salt the name of the object, and route to multiple roots, one for each salt. Each of these roots now knows the publishing node's location. When a different node attempts to retrieve the object, it can route to any of these salted roots. If one is down, another should be available.

In the basic implementation of Puddlestore, when storing an object, we only publish it from a single Tapestry node. This leads to a potential issue where the publishing node may be unreliable or may crash, leading to difficulty retrieving the published object. To combat this issue, we implemented Object Re-Replication. Rather than publishing an object from a single tapestry node, we instead publish from multiple tapestry nodes. Once we route to a root node, the root node should have multiple publishing node locations stored. If one doesn't work, we can try the others.